TEANLIS - Text Analysis for Literary Scholars

Andreas Müller[1,3]***, Markus John***[2,4]***, Jonas Kuhn***[1,3]
(1) Institut für Maschinelle Sprachverarbeitung Universität Stuttgart
(2) Institut für Visualisierung und Interaktive Systeme (VIS) Universität Stuttgart
(3) Vorname.Nachname@ims.uni-stuttgart.de
(4) Vorname.Nachname@vis.uni-stuttgart.de

Abstract

Researchers in the (digital) humanities have identified a large potential in the use of automatic text analysis capabilities in their text studies, scaling up the amount of material that can be explored and allowing for new types of questions. To support the scholars' specific research questions, it is important to embed the text analysis capabilities in an appropriate navigation and visualization framework. However, study-specific tailoring may make it hard to migrate analytical components across projects and compare results. To overcome this issue, we present in this paper the first version of TEANLIS (text analysis for literary scholars), a flexible framework designed to include text analysis capabilities for literary scholars.

1 Introduction

Researchers in the (digital) humanities have identified a large potential in the use of automatic text analysis capabilities in their text studies, scaling up the amount of material that can be explored and allowing for new types of questions. To effectively support the humanities scholars' work in a particular project context, it is not unusual to rely on specially tailored tools for feature analysis and visualizations, supporting the specific needs in the project. Support for linking up detailed technical aspects to higher-level research questions is crucial for the success of digital humanities projects, but overly study-specific tailoring limits the usability of the analysis capabilities of the tools in other projects. This effect is also in part due to the potential difficulty of separating the analysis capabilities and the visualizations used to present the results of an analysis[1].

We present the experimental text analysis framework TEANLIS (Text analysis for literary scholars), which is designed to: 1) provide text analysis capabilities which can be applied out-of-the-box and which take advantage of a hierarchical representation of text, 2) integrate functions to load documents which were processed with other text analysis frameworks (e.g. GATE (Cunningham et al., 2013)) and 3) provide standard analysis functions for documents from the recent infrastructure initiatives for the digital humanities such as CLARIN,[2] DARIAH[3] and TextGrid (Hedges et al., 2013). TEANLIS is not in and of itself meant to be a tool for literary analysis, but rather a framework which allows developers to quickly build a tool for literary analysis in particular and text analysis in general.

We work with data visualization experts involved in our project to ensure that TEANLIS can support interactive visualizations of results. Interactive visualizations enable intuitive access to the results of the computational linguistics (CL) methods we provide. TEANLIS is different from established frameworks like GATE (Cunningham et al., 2013) or UIMA (Ferrucci and Lally, 2004) in that it focuses on supporting visualizations and analysis capabilities based on a hierarchical doc-

[1]This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). Page numbers and proceedings footer are added by the organizers. License details: http://creativecommons.org/licenses/by/4.0/

[2]http://www.clarin-d.de

[3]https://de.dariah.eu/

ument structure and special analysis capabilities tailored to the needs of literary scholars. A first version of the framework is available online[4]. An example for a tool developed on the basis of TEANLIS is described in Koch et al. (2014).

The remainder of this paper is structured as follows: In section 2, we will review existing frameworks for providing literary scholars with text analysis capabilities. In section 3, we will discuss the model of document structure used for document representation in our framework. Section 4 describes a document navigation scheme implemented in the framework. Section 5 discusses how we load documents which were processed by GATE. In section 6 the different ways in which documents from TextGrid, plain text documents and plain text documents with attached structural markup can be loaded will be discussed. Section 7 describes a baseline for expression attribution, a more general version of quoted speech attribution (Pareti et al., 2013), implemented in our framework. Section 8 summarizes our contributions and discusses future work.

2 Related Work

Most similar to our framework are the widely used frameworks GATE and UIMA. Both frameworks use an offset-based format and provide a large variety of text analysis capabilities in the form of analysis components made by multiple people. To the best of our knowledge neither GATE nor UIMA represent the hierarchical structure of a document explicitly. In our framework, the hierarchical structure of a document is represented explicitly, which allows analysis capabilities based on the hierarchical structure of documents to be implemented in a straightforward manner.

Also similar to our framework are tools for making text analysis capabilities available to humanities researchers who have no background in machine learning and CL. Two of those tools are the eHumanities desktop (Gleim and Mehler, 2010) and the tool developed in Blessing et al. (2013). The eHumanities desktop is specifically designed for the needs of humanities researchers, the tool described in Blessing et al. (2013) for researchers in political science. To the best of our knowledge neither one contains facilities to represent a hierarchical representation of document structure explicitly.

The WebLicht environment (Hinrichs et al., 2010), a webservice-based orchestration facility for linguistic text processing tools, is largely orthogonal to the approach presented here, since it is centered around a classical linguistic processing chain and does not put emphasis on higher-level document navigation.

3 Model of document structure

We inherently view and analyze literary documents as having a minimal hierarchical structure. This structure consists of a linguistic and an organizational structure and forms the basis for the visualization of document structure. The smallest unit of the minimal structure is a character. Every other unit is then defined by its start and end offset in the text. For example, a token, the next largest linguistic unit in a document, is defined by its start and end offset and additional properties like its part-of-speech tag, its lemma or its relation to other tokens. Tokens are contained in sentences. Therefore characters, tokens and sentences form the minimal linguistic structure of a document.

The minimal organizational structure of a document are textual lines. Lines are the smallest unit of organizational structure. Other common units are paragraphs, pages, sub-chapters and chapters. In most cases, lines are to the organizational structure what characters are to the linguistic structure: The smallest units of organizational structure which all other units are composed of.

The hierarchical representation of documents allows for the generic implementation of a document navigation scheme which will be discussed in the next section.

4 Document navigation

The following concept for document navigation is also used in the tool based on TEANLIS mentioned earlier (Koch et al., 2014). The following example discussed with reference to figure 1 was also presented in this paper.

[4] https://github.com/AndreasMuellerAtStuttgart/TEANLIS

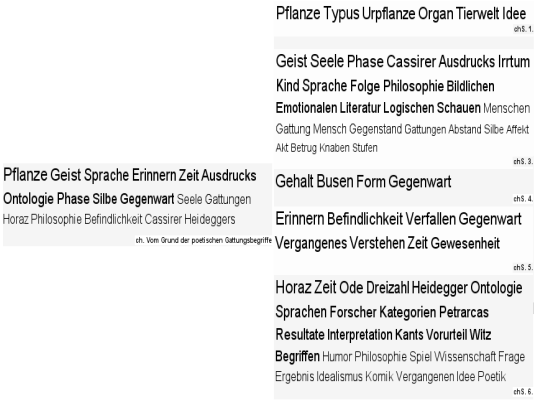

Figure 1: Example for a segmentation computed by the document navigation scheme (derived from Staiger (1946), graphic taken from Koch et al. (2014))

Word clouds are often used to give an overview over the topics occurring in a text. We provide a generic functionality for computing word clouds from the nouns occurring in the elements of a hierarchical level. For example, we can compute word clouds for every chapter in a book. This is accomplished by using each chapter as a document in a Lucene[5] index. We view each noun as a distinct term in a document and use every noun which occurs in at least one chapter as a search term. The keywords in a word cloud for a given chapter are then the nouns which scored highest when they were used as a search term for that chapter. This mechanism can be used for every level in the hierarchy. By using the lucene standard scoring formula to rank terms, we take the idf (inverse document frequency) of a term into account. Thus, the top keywords for an element on a given level of the hierarchy are not only the nouns with the highest frequency in the element. A noun which has a low frequency in the element but appears in very few or no other elements besides the given element is also likely to appear as a keyword. This follows the intuition behind the TFxIDF formula from information retrieval (Manning et al., 2008).

Some elements, like chapters, usually have multiple topics occurring in them. Those topics can sometimes be seen in a word cloud by looking for keywords which are semantically related. Ideally we would like to know where in the chapter a topic like this is discussed. Therefore, we implemented a generic functionality for splitting a hierarchy element into topic segments using the implementation of the TextTiling algorithm (Hearst, 1997) of the morphadorner library (Burns, 2013). This allows us to compute word clouds for the topic segments. In those word clouds the topics which can be seen in the chapter word cloud can be much more recognizable. An example is shown in figure 1.

Here, the words ”Erinnern” (remembering), ”Zeit” (time) and ”Gegenwart” (present) in the chapter on the left indicate a topic occurring in this chapter. The segments on the right are automatically computed and further segment the chapter. The fourth segment on the right contains those three words and also other words which are related, like ”Vergangenes” (roughly translates to ”that which was” in English). This indicates that the fourth segment on the right discusses the subtopic indicated by the three words in the segment on the left. This is an example for how the topic segmentation of arbitrary hierarchy elements can assist in document navigation.

An example for a concrete research question which can be addressed with a tool based on TEANLIS is discussed in Koch et al. (2014). This example discusses how the document navigation scheme presented in this section can help a literary scholar answer questions about which works and authors associated with different literary styles are discussed in a poetic.

5 GATE Converter

Since our framework is also offset-based converting documents in our format to GATE documents is straightforward as far as the offsets are concerned. For mapping the properties of, for example, tokens, we define a mapping from property names in our framework to the corresponding feature names in GATE. This ensures that features have the names the processing resources which are used to further process a document in GATE expect. A similar mapping is used to convert documents in GATE format to documents in our format.

A particularly useful feature of converting documents in our format to GATE documents is

[5] http://lucene.apache.org/

that it allows the generic use of the JAPE[6] system. JAPE allows the definition of regular expression grammars over annotations and their features. JAPE is easy to use, even for people who have no background in computer science. Therefore, developers can provide access to JAPE (via GATE Developer[7], the graphical interface to a lot of GATE analysis and annotation capabilities) in a simple manner, which could be very useful for literary scholars who are familiar with the JAPE system or are willing to learn it.

Note that by using the Graph Annotation Framework (GrAF) described in Ide and Suderman (2009) we could in principle also convert documents in our format to UIMA documents, because the GrAF framework supports conversion from GATE to UIMA format and back.

6 Generic xml and plain text loader

To access the documents in TextGrid in a generic way, we implemented a loader for documents in TEI (Unsworth, 2011) format and plain text documents with pre-defined structural markup. There are two types of loaders: A minimal loader and a structure-aware plain text loader.

6.1 Minimal loader

The minimal loader works for all documents in xml format which have a tag containing the text of the document. This tag has to be specified. The method simply reads the text content of the xml-element corresponding to the tag and stores it as the text of the document in our format. If possible, the language of the text is extracted from meta-data and a suitable sentence splitter and tokenizer are used to pre-process the text, giving the document at least the minimal linguistic structure.

We support six languages by integrating the OpenNLP tools[8] for sentence splitting and tokenization. The languages are: Danish, German, English, Dutch, Portugese and Swedish[9]. In the generic loader, if no line delimiting characters are given we represent the text in one-sentence-per-line format. If a document is from TextGrid we search for paragraphs by searching for <p>tags. Note that even though the minimal loader does not recognize document structure, the document navigation scheme explained before can automatically compute structure and an overview of the topics contained in the structural elements. This can be done by using topic segmentation on the whole text. Then, the resulting segments can be segmented themselves and so forth.

Note that even though the minimal loader does not recognize document structure, the document navigation scheme explained before can automatically compute structure and an overview of the topics contained in the structural elements.

6.2 Structure aware plain text loader

For text files we also provide the option of inserting structural tags to mark the standard structure our framework recognizes. To get the textual units constituting the structural elements on a given hierarchical level, we simply split the text on the structural tags provided by the user. For example, if PARAGRAPH tags are given we would simply regard all text between two PARAGRAPH tags as one paragraph. If the plain text files have structural tags which do not correspond to our tags but delimit the same units (for example, a PARAGRAPH tag given as a P tag), the user can specify a mapping between the text in the files and our tags (for example, mapping P to PARAGRAPH). This represents a simple way to attach structural markup to a text.

7 A baseline for expression attribution

We already mentioned that TEANLIS is designed to support the development of tools for literary analysis. To this end, we implemented baselines for tasks which are relevant for literary analysis. One of those tasks is expression attribution. Expression attribution is a more general type of quotation extraction (Pareti et al., 2013). For example, expression attribution includes a sentence like: ”It is, as Husserl showed, paradoxical to say they could vary.” which includes an expression of an author (Husserl). This expression is an abstract representation of the expression of Husserl, not something he actually said or wrote exactly as expressed in the sentence.

The baseline is described in detail in a paper

[6] http://gate.ac.uk/sale/tao/splitch8.html#x12-2190008
[7] http://gate.ac.uk/sale/tao/splitch3.html#x6-420003
[8] https://opennlp.apache.org/
[9] http://opennlp.sourceforge.net/models-1.5/

we submitted to STRiX 2014[10]. The paper is currently under review. Part of the following description is taken from that paper. Essentially, our technique extracts triples of the form (person, verb-cue, sentence-id). Person is the utterer of an expression, verb-cue is the verb used to detect the expression (if a verb is used to detect the expression) and sentence-id is the id of the sentence containing the expression. The baseline extracts those triples by detecting sentences which either contain the name of an author and a quotation or the name of an author and a verb indicating the presence of an expression (the verb "express" or the verb "say").

We evaluated the baseline by extracting triples from Staiger (1946). The system identified 64 instances of attributed expressions within the text. We then manually classified the instances into three classes, in order to gain insight into the behavior of the algorithm and to guide future work. If the sentence contained an attributed expression and the utterer was identified correctly, we considered the instance to be annotated fully correct. If the sentence contained an attributed expression but the utterer was not correctly identified, we considered the item to be partially correct. All other instances were considered an error. The first author of the current paper and a colleague from the same institute annotated these classes in parallel, with an initial F_1-agreement of 0.67. Differences have been adjudicated after discussion with a domain expert.

Our baseline identifies 62.5% of the utterances correctly, and for 51.6% the correct utterer was also identified.

8 Future Work

We presented a framework for developing tools to support the analysis of texts with a hierarchical structure in general and literary texts in particular. Our framework is different from the established frameworks GATE and UIMA in that it provides analysis capabilities based on the recognition of the hierarchical structure of a text. It also provides facilities for computing the hierarchical structure of arbitrary texts in a semi-automatic manner. Also, our documents can be converted to GATE documents and conversely. This allows the integration of the analysis capabilities provided by GATE. Through the GrAF framework we can theoretically convert our documents to UIMA documents, which is something we want to investigate in the future.

We plan to integrate other topic segmentation algorithms, especially algorithms for hierarchical topic segmentation like the one described in Eisenstein (2009). We are also in the process of writing genre-specific converters to convert documents from the TextGrid repository to our documents and take their existing structure into account. This can be done by recognizing predefined structural elements, like recognizing page breaks by searching for the <pb>tag. Documents from different genres are then distinguished by which of those tags they use to mark structure. This allows developers to access a large repository of literary documents without having to write converters of their own.

We are also in the process of implementing baselines for CL tasks which can benefit literary analysis. One example is the baseline for expression attribution discussed in the last section. Another type of tasks are text classification tasks like classifying paragraphs with respect to what theme they talk about. In a first baseline, themes are "aesthetic" and "poetic". Paragraphs are classified based on typical words for the themes provided by the literary scholar in our project. Although we have not formally evaluated those baselines yet we observed that they work quite well on the text in our corpus we tested them on. However, the point of implementing those baselines is to provide developers with a starting point for quickly getting results for those tasks. This enables them to see how well obvious baselines perform on their data and to assess the specific problems of the task with respect to their data.

Acknowledgments

This work has been funded by the German Federal Ministry of Education and Research (BMBF) as part of the "ePoetics" project.

[10] http://spraakbanken.gu.se/eng/strix2014

References

Andre Blessing, Jonathan Sonntag, Fritz Kliche, Ulrich Heid, Jonas Kuhn, and Manfred Stede. 2013. Towards a tool for interactive concept building for large scale analysis in the humanities. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, pages 55–64, Sofia, Bulgaria, August. Association for Computational Linguistics.

Philip R. Burns. 2013. Morphadorner v2: A java library for the morphological adornment of english language texts. October.

Hamish Cunningham, Valentin Tablan, Angus Roberts, and Kalina Bontcheva. 2013. Getting more out of biomedical documents with gate's full lifecycle open source text analytics. *PLOS Computational Biology*.

Jacob Eisenstein. 2009. Hierarchical text segmentation from multi-scale lexical cohesion. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 353–361, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Ferrucci and Adam Lally. 2004. Uima: An architectural approach to unstructured information processing in the corporate research environment. *Nat. Lang. Eng.*, 10(3-4):327–348, September.

Rüdiger Gleim and Alexander Mehler. 2010. Computational linguistics for mere mortals — powerful but easy-to-use linguistic processing for scientists in the humanities. In *Proceedings of LREC 2010*, Malta. ELDA.

Marti A. Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.*, 23(1):33–64, March.

Mark Hedges, Heike Neuroth, Kathleen M. Smith, Tobias Blanke, Laurent Romary, Marc Kster, and Malcolm Illingworth. 2013. Textgrid, textvre, and dariah: Sustainability of infrastructures for textual scholarship. *Journal of the Text Encoding Initiative [Online]*, June.

Erhard W. Hinrichs, Marie Hinrichs, and Thomas Zastrow. 2010. WebLicht: Web-Based LRT Services for German. In *Proceedings of the ACL 2010 System Demonstrations*, pages 25–29.

Nancy Ide and Keith Suderman. 2009. Bridging the gaps: Interoperability for graf, gate, and uima. In *Proceedings of the Third Linguistic Annotation Workshop*, ACL-IJCNLP '09, pages 27–34, Stroudsburg, PA, USA. Association for Computational Linguistics.

Steffen Koch, Markus John, Michael Wörner, Andreas Müller, and Thomas Ertl. 2014. Varifocalreader - in-depth visual analysis of large text documents. In *To appear in: IEEE Transactions on Visualization and Computer Graphics (TVCG)*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Silvia Pareti, Timothy O'Keefe, Ioannis Konstas, James R. Curran, and Irena Koprinska. 2013. Automatically detecting and attributing indirect quotations. In *EMNLP*, pages 989–999. ACL.

Emil Staiger. 1946. *Emil Staiger: Grundbegriffe der Poetik*. Atlantis Verlag Zürich.

John Unsworth. 2011. Computational work with very large text collections. *Journal of the Text Encoding Initiative [Online]*.